(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
14 juillet 2005 (14.07.2005)

PCT

(10) Numéro de publication internationale
WO 2005/064459 A2

(51) Classification internationale des brevets[7] : G06F 9/445

(21) Numéro de la demande internationale :
PCT/FR2004/003353

(22) Date de dépôt international :
22 décembre 2004 (22.12.2004)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
0315487 24 décembre 2003 (24.12.2003) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **TRUSTED LOGIC** [FR/FR]; 5, rue du Bailliage, F-78000 Versailles (FR).

(72) Inventeur; et

(75) Inventeur/Déposant *(pour US seulement)* : **VETILLARD, Eric** [FR/FR]; Résidence Garbejaire 2, Bât 1, 1, passage des Pignes, F-06560 Valbonne (FR).

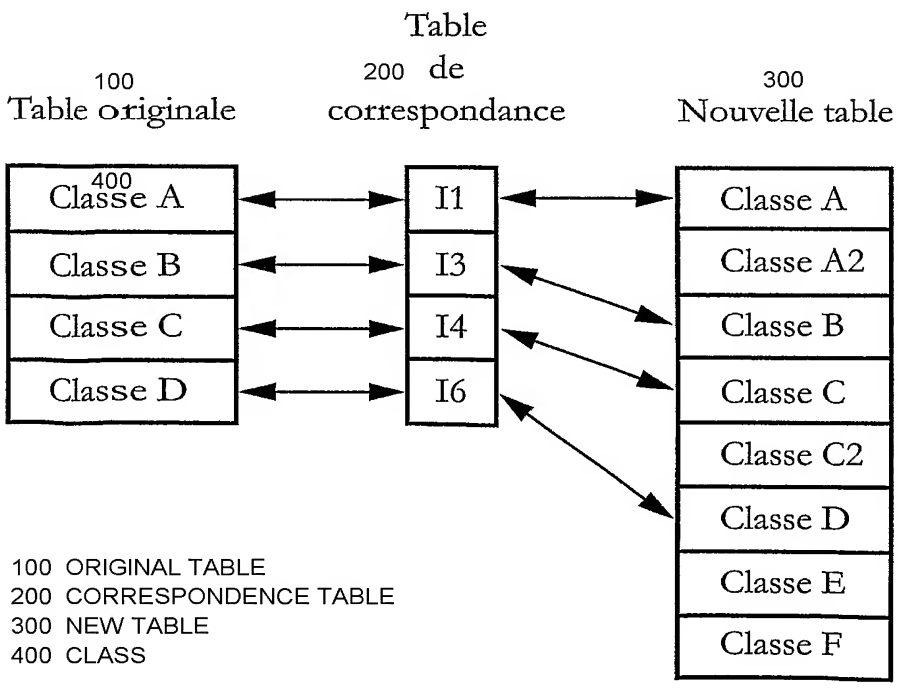
(74) Mandataire : **DE SAINT PALAIS, Arnaud**; Cabinet Moutard, 35, rue de la Paroisse, F-78000 Versailles (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

[Suite sur la page suivante]

(54) **Title:** METHOD FOR UPDATING APPLICATIONS FOR A CHIP CARD

(54) **Titre :** PROCEDE DE MISE A JOUR D'APPLICATIONS POUR CARTE A PUCE.



(57) Abstract: The invention relates to a method for enabling a new version of an application to be loaded onto a computer processing device. According to said method, information on the correspondence (I1, I3, I4, I6) between the classes (A to D) of the old version of the application and the classes (A to F) of the new version of the application, and information about correspondence between the static fields of the old version of the application and static fields of the new version of the application, is calculated prior to the loading. Said correspondence information is then associated in order to modify the objects in such a way that they point towards classes of the new version and use the new identifiers of the static fields of the new version of the application.

(57) Abrégé : Le procédé selon l'invention permet le chargement sur un dispositif informatique d'une nouvelle version d'une application. Il consiste à calculer, préalablement à ce chargement d'une part une information de correspondance (I1, I3, I4, I6) entre les classes (A à D) et (A à F) de l'ancienne version de l'application et de la nouvelle version de l'application et d'autre part, une information de correspondance entre les champs statiques de l'ancienne version et de la nouvelle version de l'application, puis à associer ces informations de correspondance pour modifier les objets de façon à ce qu'ils pointent vers les classes de la nouvelle version et qu'ils utilisent les nouveaux identificateurs des champs statiques de la nouvelle version de l'application.

WO 2005/064459 A2

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclaration en vertu de la règle 4.17 :

— *relative à la qualité d'inventeur (règle 4.17.iv)) pour US seulement*

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

PROCEDE DE MISE A JOUR D'APPLICATIONS POUR CARTE A PUCE.

La présente invention concerne un procédé de mise à jour d'applications pour carte à puce.

Elle a plus particulièrement pour objet de permettre le chargement d'une nouvelle version d'une application dans une carte à puce tout en conservant les données utilisées par les précédentes versions de l'application.

La plupart des systèmes de cartes à puce actuelles utilisent des machines virtuelles, la plus populaire d'entre elle étant celle de « Java Card » (marque déposée par la Société Sun Microsystems). Dans ce système, les données persistantes des applications sont stockées comme des objets, et en particulier comme des instances des classes définies dans les programmes chargés dans la carte. La nature persistante de ces objets rend donc les programmes actifs en permanence dans la carte, à l'inverse de ce qui se produit dans les systèmes classiques (stations de travail, ordinateurs de bureau). Cela pose un problème particulier pour la mise à jour des programmes, puisqu'il n'est pas possible de profiter d'un moment où le programme n'est pas actif pour le mettre à jour.

De plus, la plupart des plates-formes utilisent un format de code binaire optimisé (format dit « CAP File » dans le cas de « Java Card »). Ce format permet de ne charger sur une carte que les informations strictement nécessaires à l'exécution du programme. En particulier, il ne contient pas

toujours les informations nécessaires pour effectuer une mise à jour de l'application.

Enfin, une carte à puce est un contexte d'exécution très particulier, dans la mesure où elle est diffusée en de très nombreux exemplaires, et elle contient souvent des informations confidentielles. En particulier, il est très difficile d'envisager de mettre à jour une application en effaçant l'application existante et ses données, en rechargeant la nouvelle application, puis en rechargeant les données. En effet, les données d'initialisation des applications sont généralement très sensibles, et ne doivent absolument pas être manipulées en dehors des sites de production des cartes, sous des conditions de sécurité très contrôlées.

La nature des problèmes rencontrés dépend fortement des caractéristiques de la mise à jour souhaitée. Pour la mise à jour d'une application, les problèmes sont les suivants :

- Mise à jour de code simple.
  Le problème est ici de corriger des défauts du programme, sans en modifier la structure. Il s'agit donc simplement de modifier la définition de certaines méthodes, et éventuellement de rajouter de nouvelles méthodes, voire de nouvelles classes (en dehors de la hiérarchie existante).

- Mise à jour de code avec modification de la hiérarchie de classes.
  Le problème est ici de corriger des défauts structurels du programme, nécessitant une modification de la hiérarchie de classes (habituellement par insertion d'une classe au milieu d'une hiérarchie existante pour prendre en compte un comportement spécifique).

- Mise à jour avec modification de la structure de l'objet.
  Le problème est ici de corriger un défaut nécessitant le stockage d'informations supplémentaires, et donc l'ajout de champs de données dans des classes existantes. Ce problème est plus complexe dans la mesure où les objets existants doivent être modifiés pour prendre en compte les modifications.

Dans certains cas, l'application à mettre à jour est accessible depuis d'autres applications, en particulier quand l'application exporte des interfaces partageables et quand l'application est en fait une librairie. D'un point de vue purement technique, de telles mises à jour posent les mêmes problèmes que les mises à jour simples d'applications, à la différence près que les mises à jour doivent être appliquées à toutes les applications qui importent les fonctions modifiées.

L'invention a donc plus particulièrement pour but la réalisation d'un mécanisme de chargement d'une nouvelle version d'une application, qui remplace une application déjà présente sur une carte avec des garanties en matière de sécurité et de continuité de fonctionnement, ce mécanisme permettant donc de modifier des applications sans interrompre leur fonctionnement.

Elle se propose d'aboutir à ce résultat en utilisant une option spécifique de commande de chargement et un format de chargement compatible avec celui défini dans la spécification « Java Card ».

En vue de parvenir à ces résultats, elle propose, d'une façon générale, un procédé pour le chargement sur un dispositif informatique d'une nouvelle version d'une application dans un langage de programmation à objets et permettant, notamment, l'introduction de classes supplémentaires, la

modification de la hiérarchie de classes et la définition de champs et de méthodes supplémentaires.

Selon l'invention, ce procédé consiste à :

- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les classes de l'ancienne version de l'application et les classes de la nouvelle version de l'application ;
- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les identificateurs des champs statiques de l'ancienne version de l'application des identificateurs des champs statiques de la nouvelle version de l'application ;
- associer lesdites informations de correspondance à la nouvelle version de l'application chargée sur le dispositif ;
- utiliser lesdites informations de correspondance pour modifier les objets de façon à ce qu'ils pointent vers les classes de la nouvelle version de l'application et qu'ils utilisent les nouveaux identificateurs des champs statiques de la nouvelle version de l'application.

Avantageusement, lesdites informations de correspondance pourront consister en des tables de correspondance.

Elles pourront être omises dans le cas où les modifications des objets ne sont pas nécessaires, par exemple, et de façon non limitative, quand aucune classe supplémentaire n'est ajoutée dans la nouvelle version de l'application ou quand les classes ajoutées ne modifient pas la hiérarchie de classes.

Le procédé selon l'invention pourra comprendre l'exécution de méthodes de mise à jour des données de l'application après l'installation de la nouvelle version de l'application.

Un mode d'exécution de l'invention sera décrit ci-après, à titre d'exemple non limitatif, avec référence aux dessins annexés dans lesquels :

Les figures 1a et 1b sont des tableaux comparatifs montrant la hiérarchie de classes de l'application, dans sa version originale (figure 1a) et dans sa nouvelle version (figure 1b) ;

La figure 2 est une table de correspondance donnant, pour chaque classe de l'application originale, l'index de la classe correspondante dans la nouvelle application ;

Les figures 3a et 3b sont des tableaux comparatifs montrant une hiérarchie avec la propriété recherchée, hiérarchie originale (figure 3a) et nouvelle hiérarchie (figure 3b) ;

La figure 4 est une table de correspondance entre des tables de correspondance (table originale/nouvelle table) correspondant à la hiérarchie illustrée figure 3b ;

La figure 5 est une table de correspondance du type de celle de la figure 4 pour les champs statiques ;

Les figures 6 à 8 sont des représentations schématiques illustrant les états d'une carte à puce, avant mise à jour (figure 6), après chargement de la nouvelle application (figure 7) et après modification des objets (figure 8).

La mise en oeuvre de l'invention se fait en plusieurs étapes :

- Préparation du fichier de chargement.
- Chargement du fichier et édition de liens.

- Mise à jour des objets de l'application.
- Exécution d'une méthode spécifique de mise à jour.

Le fichier de chargement doit contenir des informations spécifiques qui permettent d'établir la correspondance avec une ou plusieurs versions antérieures de l'application. Afin de générer le fichier de mise à jour, il est nécessaire de disposer des objets suivants :

- Tous les fichiers de classe de l'application originale.
- Le fichier de chargement (« CAP File » dans le cas de « Java Card ») de l'application originale.
- Tous les fichiers de classe de la nouvelle version de l'application.
- Tous les fichiers d'export requis pour construire la nouvelle version de l'application.

Si le fichier de chargement de l'application originale comprend des informations de typage et de nommage des classes, il n'est alors pas indispensable de disposer des fichiers de classe de l'application originale.

Les données d'entrées doivent respecter les contraintes suivantes :

- Pour chaque élément de l'application originale, il existe dans la nouvelle version de l'application un élément équivalent (de même nom et de même type).
- Si l'application originale exporte une interface externe aux autres applications, cette interface externe doit être inchangée dans la nouvelle version de l'application.
- Les fichiers d'export utilisés dans la nouvelle version de l'applications sont binaires-compatibles (c'est-à-dire qu'ils peuvent être reliés sans modification aux autres parties de l'application initiale) avec ceux utilisés dans l'application originale. Dans le cas de « Java Card », il s'agit de ceux

qui sont listés dans le composant d'import du fichier de chargement original.

Le fichier de chargement généré est un fichier de chargement normal, qui peut donc être chargé sur n'importe quelle carte. Ce fichier contient un composant optionnel avec les informations suivantes pour chaque version précédente de l'application prise en considération :

- le numéro de cette version,
- une table qui établit une correspondance entre chaque classe ou interface définie dans l'ancienne version et la nouvelle version de cette classe ou interface dans la nouvelle version de l'application,
- une table qui établit une correspondance entre l'identifiant de chaque champ statique de l'ancienne version et le nouvel identifiant de ce champ dans la nouvelle version.

Ces informations supplémentaires ne sont nécessaires que pour les opérations de mise à jour. Le même fichier binaire peut donc servir pour le chargement de l'application et pour la mise à jour d'applications.

Dans un premier exemple, les hiérarchies de classe de l'application dans sa version originale et dans sa nouvelle version sont montrées sur les figures 1a et 1b. La hiérarchie originale comporte quatre classes (Classe A à Classe D) et la nouvelle hiérarchie comporte quatre classes supplémentaires (Classe E, Classe F, Classe A2, Classe C2) dont certaines (Classe A2 et Classe C2) sont insérées dans la hiérarchie originale.

Dans ce cas, une table de correspondance peut être établie.

Comme illustré figure 2, cette table de correspondance peut, par exemple, donner, pour chaque classe de l'application originale, l'index I1, I3, I4, I6 de la classe correspondante dans la nouvelle application.

L'exemple suivant concerne un cas particulier dans lequel la table supplémentaire n'a pas besoin d'être incluse dans le fichier de chargement. Ceci est possible si :

- La hiérarchie des classes de l'application originale est conservée inchangée dans la nouvelle version de l'application (aucune classe n'est insérée dans la hiérarchie).

- Les classes de l'application originale sont définies en premier dans le nouveau ficher de chargement, et dans le même ordre que dans le fichier de chargement original.

Les figures 3a et 3b montrent une hiérarchie avec la propriété recherchée dans laquelle la nouvelle hiérarchie (figure 3b) inclut les classes E et F sans insertion dans la hiérarchie originale (figure 3a) des classes A à D.

La figure 4 montre les tables de classes correspondant à cette hiérarchie, et respectant la propriété d'ordre indiquée ci-dessus. On voit alors que la table de correspondance est triviale et n'est pas nécessaire.

Le fichier doit également contenir une table de correspondance entre les champs statiques de l'application originale et ceux de la nouvelle version de l'application. Cette table est similaire à la précédente, mais elle est cette fois indexée par les identifiants des nouveaux champs statiques ; les éléments de la table qui correspondent à des nouveaux champs contiennent un identifiant invalide (I0 dans l'exemple), et les autres éléments contiennent l'identifiant du même champ dans la version originale.

La figure 5 montre un exemple d'une telle table comprenant :

- une table originale comportant les champs A.champ1, A.champ2, C.champ1, A.champ3,

- une nouvelle table comportant les champs A.champ2, A.champ1, C.champ1, C.champ2, A.champ3, F.champ1,

- une table de correspondance dans laquelle le champ A.champ1 est indexé par l'identifiant I1, A.champ2 est indexé par l'identifiant I2, C.champ1 est indexé par l'identifiant I3, A.champ3 est indexé par l'identifiant I4 ; les nouveaux champs C.champ2 et F.champ1 contiennent un identifiant invalide = I0.

Une fois le fichier approprié généré, il faut le charger dans la carte. On peut supposer que l'état de la carte avant chargement est comme indiqué en figure 6 avec un objet Obj1 en ClasseA et un objet Obj2 en ClasseB de l'application App1.

Le chargement s'effectue en utilisant la procédure d'édition de liens normale du système.

La figure 7 montre l'état de la carte après le chargement de la nouvelle application. La nouvelle version de l'application App1' est chargée, mais les objets Obj1, Obj2 de l'application pointent toujours vers l'ancienne version (ClasseA, ClasseB de l'application App1).

Une des phases du chargement est l'initialisation des champs statiques. La procédure standard d'initialisation est appliquée, sauf pour les champs qui sont hérités de l'application originale. Pour ces champs, la valeur initiale définie

dans la nouvelle version de l'application est ignorée, et l'ancienne valeur est copiée.

L'étape suivante consiste donc à modifier les liens des objets pour qu'ils pointent vers les nouvelles classes. Il faut alors parcourir tous les objets de l'application et utiliser la table de correspondance entre anciennes et nouvelles classes pour identifier la nouvelle classe de l'objet.

Le résultat est montré figure 8, où les objets Obj1, Obj2 pointent sur les nouvelles classes (ClasseA', ClasseB').

Lors de cette étape, il se peut que les objets doivent être modifiés, si de nouveaux champs ont été ajoutés à leur classe. Dans ce cas, un nouvel objet (avec les nouveaux champs) est alloué, les valeurs des anciens champs sont copiées de l'ancienne version de l'objet, et les valeurs des nouveaux champs sont initialisées à leur valeur par défaut (0 pour les entiers, "false" pour les booléens, et "null" pour les pointeurs). Les références vers l'ancien objet sont ensuite mises à jour en utilisant des techniques classiquement implémentées dans les récupérateurs de mémoire.

La dernière étape consiste à laisser l'application faire toutes les opérations nécessaires à la mise à jour des données et, en particulier, de procéder à l'initialisation des nouveaux champs (champs statiques, ou champs insérés dans des objets). Les applets qui ont besoin d'effectuer une mise à jour doivent implémenter une interface spécifique, dans laquelle une méthode est définie. La mise à jour consiste donc à parcourir la table des applets enregistrées dans le système et, pour chaque applet qui est une instance d'une classe définie dans le package mis à jour et qu'implémente l'interface de mise à jour, d'invoquer la méthode avec les paramètres appropriés.

Revendications

1. Procédé pour le chargement sur un dispositif informatique d'une nouvelle version d'une application dans un langage de programmation à objets et permettant, notamment, l'introduction de classes supplémentaires, la modification de la hiérarchie de classes et la définition de champs et de méthodes supplémentaires,
caractérisé en ce qu'il consiste à :

- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les classes de l'ancienne version de l'application et les classes de la nouvelle version de l'application ;
- calculer préalablement à ce chargement une information permettant d'établir la correspondance entre les identificateurs des champs statiques de l'ancienne version de l'application des identificateurs des champs statiques de la nouvelle version de l'application ;
- associer lesdites informations de correspondance à la nouvelle version de l'application chargée sur le dispositif ;
- utiliser lesdites informations de correspondance pour modifier les objets de façon à ce qu'ils pointent vers les classes de la nouvelle version de l'application et qu'ils utilisent les nouveaux identificateurs des champs statiques de la nouvelle version de l'application.

2. Procédé selon la revendication 1,
caractérisé en ce que lesdites informations de correspondance sont des tables de correspondance.

3. Procédé selon l'une des revendications 1 et 2,
caractérisé en ce que lesdites informations de correspondance sont omises dans le cas où les modifications des objets ne sont pas nécessaires quand aucune classe supplémentaire n'est ajoutée dans la nouvelle version de

l'application ou quand les classes ajoutées ne modifient pas la hiérarchie de classes.

4. Procédé selon l'une des revendications précédentes,
caractérisé en ce qu'il comprend l'exécution de méthodes de mise à jour des données de l'application après l'installation de la nouvelle version de l'application.

5. Procédé selon l'une des revendications précédentes,
caractérisé en ce que ledit dispositif informatique est une carte à puce.

6. Procédé selon l'une des revendications précédentes,
caractérisé en ce que ledit langage de programmation est le langage « Java Card ».

Figure 1a

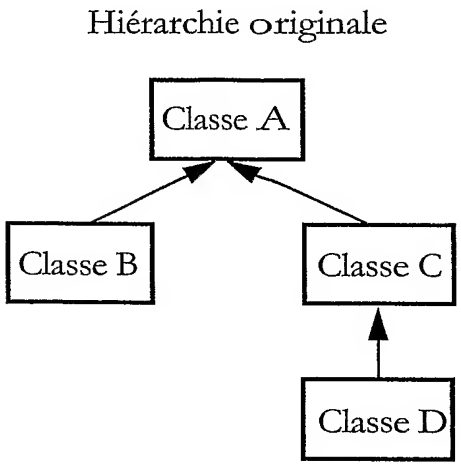


Figure 1b

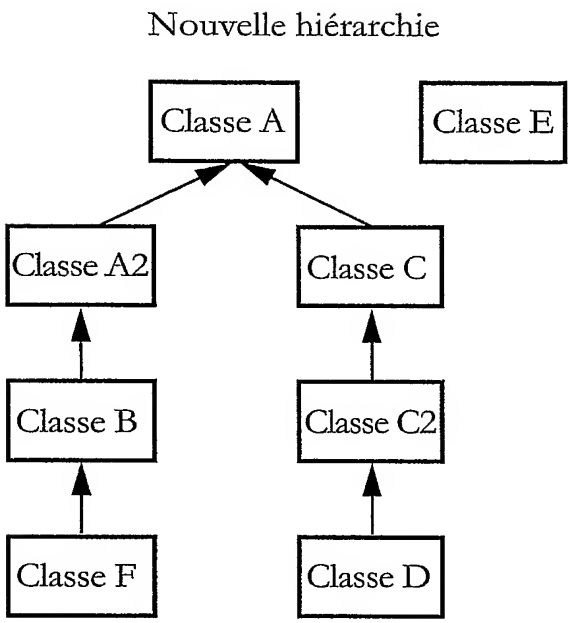


Figure 2

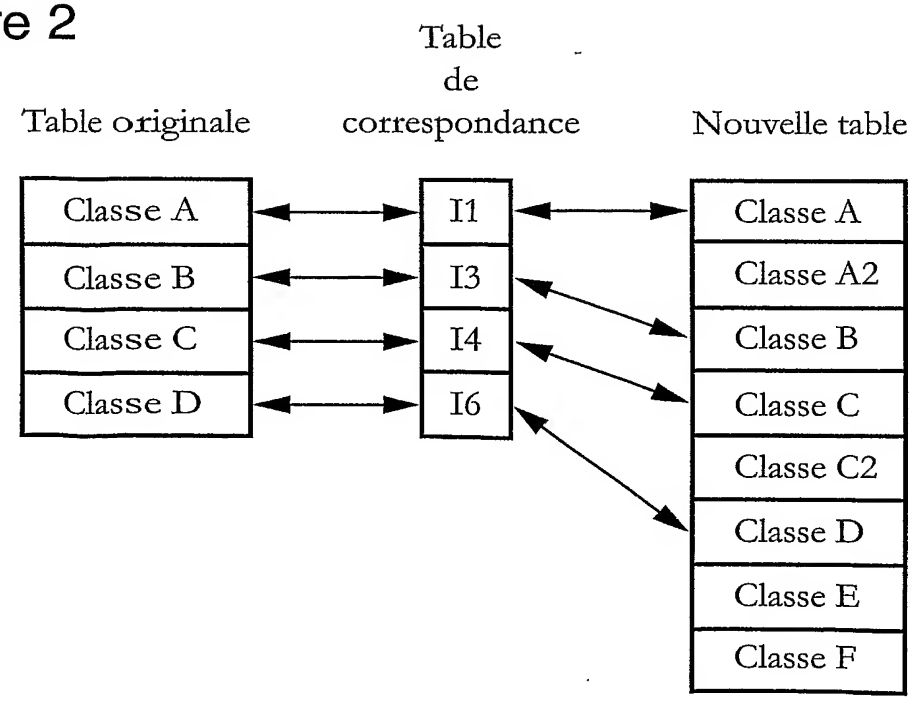


Figure 3a

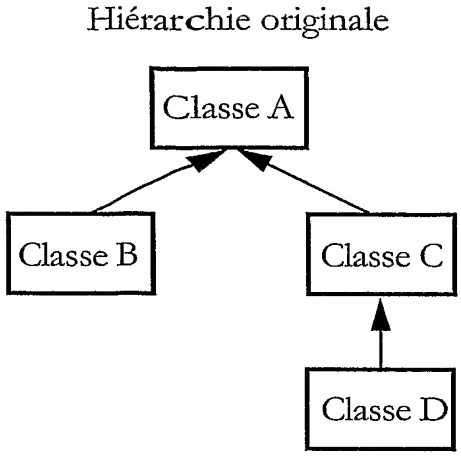


Figure 3b

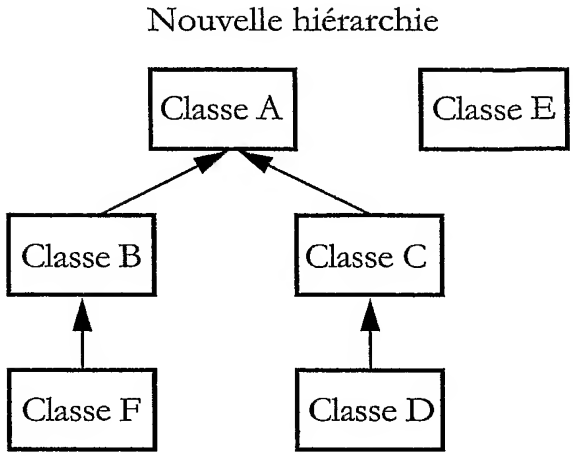


Figure 4

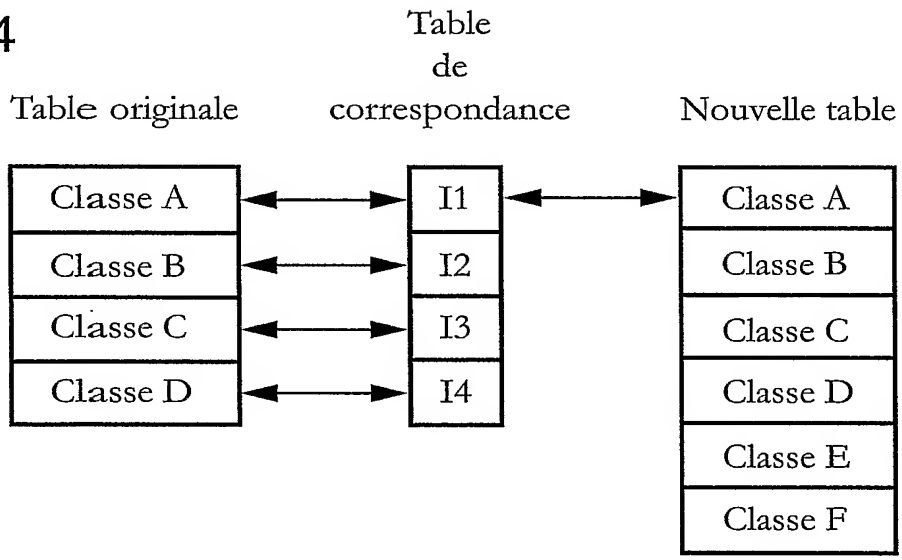


Figure 5

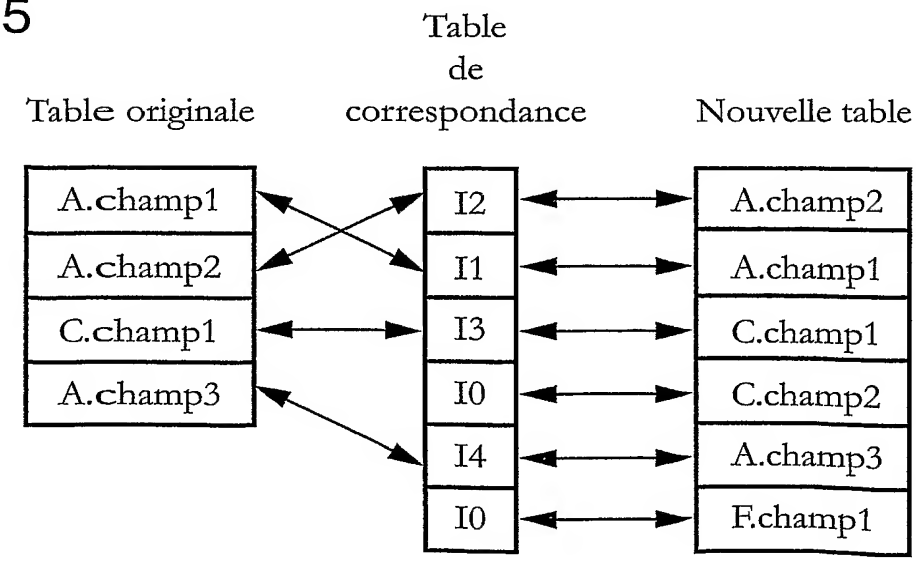


Figure 6

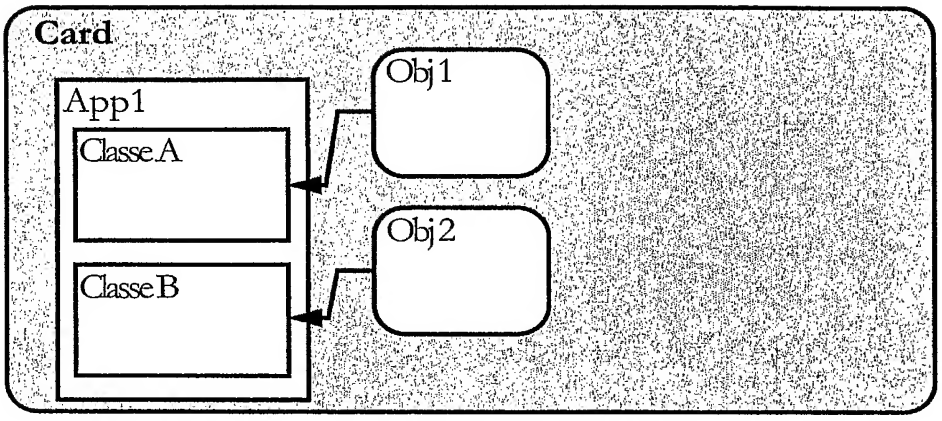


Figure 7

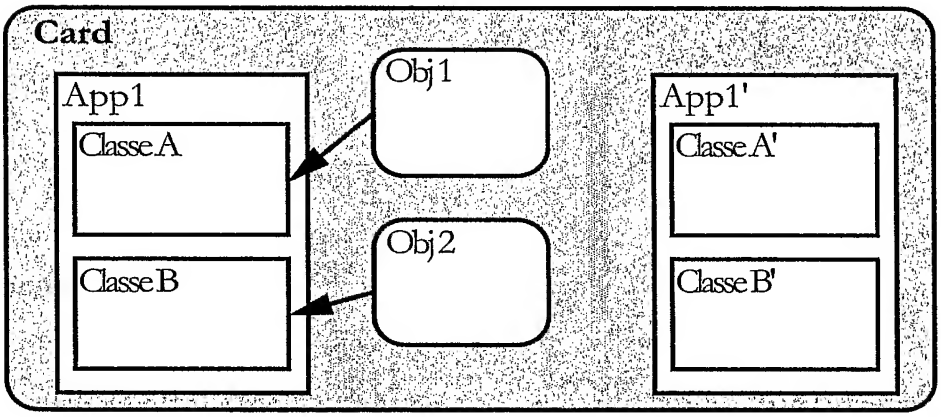


Figure 8